UNITED STATES PATENT AND TRADEMARK OFFICE

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/527,812	11/29/2005	Christophe Justin Evrard	550-619	4576

23117 7590 12/08/2009
NIXON & VANDERHYE, PC
901 NORTH GLEBE ROAD, 11TH FLOOR
ARLINGTON, VA 22203

EXAMINER	
VICARY, KEITH E	

ART UNIT	PAPER NUMBER
2183	

MAIL DATE	DELIVERY MODE
12/08/2009	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

RECORD OF ORAL HEARING

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE BOARD OF PATENT APPEALS

AND INTERFERENCES

EX PARTE CHRISTOPHE JUSTIN EVRARD and JULIE-ANNE FRANCOISE MARIE PRUVOST

Appeal 2009-005011
Application 10/527,812
Technology Center 2100

Oral Hearing Held: November 4, 2009

Before JOSEPH L. DIXON, JEAN R. HOMERE, and JAMES R. HUGHES, *Administrative Patent Judges.*

APPEARANCES:

ON BEHALF OF THE APPELLANTS:

STANLEY C. SPOONER, ESQUIRE
Nixon & Vanderhye, P.C.
901 North Glebe Road, 11th Floor,
Arlington, VA 22203

The above-entitled matter came on for oral hearing on Wednesday, November 4, 2009, at The U.S. Patent and Trademark Office, 600 Dulany Street, Alexandria, Virginia, before Victor Lindsay, Notary Public.

THE USHER: Calendar No. 20, Appeal No. 2009-5011. Mr. Spooner?

JUDGE DIXON: Hello, Mr. Spooner.

MR. SPOONER: Good morning.

JUDGE DIXON: You have 20 minutes. You may begin whenever you're ready.

MR. SPOONER: Okay, let me give him --

THE REPORTER: Thank you very kindly, sir.

MR. SPOONER: Just to make sure.

Well, I'm here this morning representing Advanced Risk Machines, Limited, ARM, a UK computer design company, and the problem that this invention is addressing is a problem with respect to security of computer interconnections, smart cards and that sort of thing. It appears that by observing the external timing and power requirements of a particular program, you can figure out what some of the algorithms are in that program, what the instructions are, what the private key might be. So to avoid that, what people came up with is a way to if it says write this particular information to storage, you would write it to a dummy store or to a regular store so that someone looking at the external power requirements couldn't see any difference. So they would not be able to know that aha, when I put this input in, it's doing a write at this point; therefore it's probably doing this thing.

So with such a system, it will defeat what they call the simple power analysis, SPA. Simple power analysis uses an algorithm, and depending upon the branching, it branches to a proper storage and writes to that one, or it branches to a dummy storage, a trash register, and writes to that one. Unfortunately, that branching that occurs, you can tell that. If you look at detailed power, time traits of a particular circuit over a number of times, the so-called differential power analysis, if you look at that over a number of cycles, you can deduce where the branching is occurring. And so it's deducing that branching that gives you the information as to what the key is for that particular secured cryptographic program.

All right, Applicant came up with a way to avoid having that branching attribute being discernible from the external power and timing characteristics of the circuit. The claims specify it. The last two items of the claim, the interrelationship, at least one data processing instruction executed by the processor core is a conditional-write data processing instruction. We'll just call that instruction to make things simple. That instruction encodes condition codes, so that interrelationship in terms of the data processing register and the -- and that instruction, that's critical.

The second critical thing is having a trash register. Now trash registers are known in the past. They've had them before. The QIU reference has a trash register. However, it doesn't have a trash register that has the interrelationships specified in the claim. In other words, one, to which a result data value will be written instead of a data processor register. In other words, that's dependent on this condition code that's contained in the instruction. Remember the QIU reference, just like the other SPA resistant programs, are directed to an algorithm that has branching right at this place or right at this place. The invention has this conditional-write data processing instruction, and encoded within that is this condition code that says okay, you're permitted to write it, or you're not permitted to write it.

JUDGE HOMERE: Counselor?

MR. SPOONER: Yes, sir.

JUDGE HOMERE: Okay, well, let me stop you right there and direct your attention to QIU at column 3, the last paragraph, where it's talking about conditional processes. It's discussing a process of performing an unnecessary multiplication algorithm, multiplication operation.

MR. SPOONER: Sure, absolutely.

JUDGE HOMERE: And then if the result is a zero, therefore the key value would be -- the result would be stored in the dummy register.

MR. SPOONER: Absolutely.

JUDGE HOMERE: Yeah, and if it's a one, then it would be stored in the data processing register.

MR. SPOONER: Okay.

JUDGE HOMERE: Okay. How is that different from what you just said, because that -- the storing of that results or in the QIU is conditional upon the value of the -- upon it being a zero or one?

MR. SPOONER: Okay. Firstly, it's not an instruction. It's not a write instruction, and it's not something that's encoded within that write -- that conditional write data instruction. Secondly --

JUDGE HOMERE: Are you saying that the multiplication operation is not an instruction?

MR. SPOONER: No, that's not -- that's part of the algorithm. That's part of the security algorithm that you're trying to break or that somebody is trying to break.

JUDGE HOMERE: No, it's the system that performed that multiplication algorithm in order to fool the attacker or someone, yeah, in order to obstruct --

MR. SPOONER: Absolutely correct.

JUDGE HOMERE: Okay.

MR. SPOONER: That's exactly what the prior art is trying to do. Both systems are trying to fool the attacker.

JUDGE HOMERE: Exactly.

MR. SPOONER: QIU or QIU, I'm not sure how you pronounce it, is a system that uses the old method. It has an algorithm, it's an algorithm based system, and it has branching which occurs based on the algorithm, so you can then -- you put a bunch of inputs into this to try to break it and see what happens and measure the power and timing requirements, and you will be able to detect over a large number of cycles the branching effect, when it branches, when it doesn't branch. You can detect that.

JUDGE HOMERE: Exactly.

MR. SPOONER: The way the Applicant's claim invention works to defeat even that is it has the same trash register, it has the same writing, but how it does that writing is the difference, and it has this programming which -- well, it's not programming. It's part of the machine, but it has this conditional-write data processing instructions in there. In that instruction, encoded in the instruction is a condition code that says yeah, this is real data or no, this isn't real data, and that's the thing that tells it where to write and the fact --

JUDGE HOMERE: Yeah --

MR. SPOONER: Let me just finish that one thought.

JUDGE HOMERE: Sure.

MR. SPOONER: The fact that that is encoded within the instruction means that that can't be detected externally. It's going to see that. It's going to write. It just writes to one place or the other, but it doesn't make that branching decision in the algorithm which can be sensed externally.

JUDGE DIXON: So it's more of a random write?

MR. SPOONER: Yeah, well, it writes all the time, so it's --

JUDGE DIXON: Right.

MR. SPOONER: So if you're looking at this thing and you're seeing oh, this is the power and the time required to write, it does that every time, and only the internal machine, because this condition code is coded in that instruction, only the machine sees aha, for the first one we want to put it in the active data store. The second one and the third one, we just dump those to the dummy register. But externally, the power output or the power usage and the timing usage of that thing is --

JUDGE DIXON: It's going to be the same or similar.

MR. SPOONER: You can't see it. You can't see any distinguishable difference.

JUDGE HOMERE: I thought that was the idea behind the unnecessary multiplication operation because QIU -- hold on a second. Let me finish that thought.

MR. SPOONER: You're absolutely correct.

JUDGE HOMERE: Yeah, QIU is performing and is continuously performing those multiplication operations, these unnecessary multiplication operations in order to generate those zeros or ones.

MR. SPOONER: Absolutely, yeah.

JUDGE HOMERE: Okay, and then the -- I mean if you have a zero, then it goes into the trash register, or one, it goes to the data processing register. Now aren't these conditions that are within the -- that are part of a result of that operation, the multiplication operations?

MR. SPOONER: No, no. They're like a flag or a private key, a bit in a private key or something like that. That causes the algorithm to branch one way or the other.

JUDGE HOMERE: Yes.

MR. SPOONER: Which then says where it's going to be stored. So they're all the same as far as trying to -- QIU is trying to do the dummy operations as well, but when it says do the dummy operation, it gives a signature of that decision.

JUDGE HOMERE: Yeah, but --

MR. SPOONER: Because it's not encoded within a conditional write instruction. It's part of the algorithm. So that's the difference. The -- ours is not an algorithm based system. It's a write instruction and hidden in the write instruction, encoded within the instruction, as the claim says --

JUDGE HOMERE: And as the Examiner submitted in the Examiner's Answer, well, an algorithm is made up of a variety of instructions, like as the

Examiner said well, said the multiplication algorithm has -- is made up of a variety of instructions where you have well, add this, multiply there, store there. These are instructions.

MR. SPOONER: Um-hum..

JUDGE HOMERE: Okay, and then one of which being well, if -- and one of these instructions being if the value -- if the result value is a zero, store in the register, otherwise store value in the trash register.

MR. SPOONER: Right.

JUDGE HOMERE: If it is a one, then store in the data processing register.

MR. SPOONER: Right.

JUDGE HOMERE: I don't understand why this would not be instructions that are conditional -- as a matter of fact, QIU refers to them as conditional processes, okay, in the section, in the section that I just pointed you -- directed you to.

MR. SPOONER: All right, but conditional isn't the word. The operative language is -- look up in the claim, at least one data processing instruction and -- is a conditional write data processing instruction encoding condition codes specifying conditions under which blah, blah, blah. It will or will not be permitted to write data to effect a change in state.

So QIU doesn't have any conditional write instruction. It doesn't have any instruction that has an encoded condition code and that -- and because it doesn't have that, it can't possibly have the interrelationship that that encoded condition code says to write or not write to the processor itself, processor store as opposed to the trash store. So it has some similarities --

JUDGE HOMERE: So counselor, if I -- let me see if I understand you well.

MR. SPOONER: Okay.

JUDGE HOMERE: So you're saying -- so you're agreeing that QIU teaches if the result of that multiplication is a one, store in the data processing system. If it's a zero, store in the trash register. So we're saying that when it's a zero --

MR. SPOONER: No, that's incorrect I believe. Look at column 3, lines 53 to 56.

JUDGE HOMERE: Okay.

MR. SPOONER: For example, when the value of a bit in the private key is a one --

JUDGE HOMERE: Okay.

MR. SPOONER: -- an associated multiplication operation occurs. When the value of the bit in the key is zero, no associated multiplication occurs. You're talking about a different operation. We're not talking about a write instruction and then something encoded in that write instruction that says yes, this is the data processing store, or this is the trash register. That part of QIU relates to whether or not a multiplication process is to occur. That's part of the algorithm.

JUDGE HOMERE: Yes, but there's also a decision that's made between where the data itself -- whether the data should be stored in the dummy register or the data processing register as well. I think that's spelled out in column 6, line --

MR. SPOONER: Oh, that's a different --

JUDGE HOMERE: What is it?

MR. SPOONER: You were talking about column 3. I'm sorry. I misunderstood.

JUDGE HOMERE: Yeah, oh, yeah. That further -- column 6, lines 5 through 15 expands --

MR. SPOONER: Yes.

JUDGE HOMERE: -- further upon the concept of the multiplication of the zero and the ones, and if you look at from column 6, lines 5 --

MR. SPOONER: But again --

JUDGE HOMERE: -- it says that if the value key bit is a one indicating a modular multiplication, okay, and the result is stored in register 512.

MR. SPOONER: Sure, yeah.

JUDGE HOMERE: Uh-huh.

MR. SPOONER: Continue down.

JUDGE HOMERE: Okay, continue down, and then if it's a zero it's stored in register 516 --

MR. SPOONER: No, no.

JUDGE HOMERE: -- which is the trash register.

MR. SPOONER: No, no, it says if it's a zero, "no modular multiplication is necessary; yet to fool the attacker, the normal multiplication modular reduction store to memory process is implemented."

JUDGE HOMERE: Is implemented and then go -- read the following sentence. So the use of register 516, which is the dummy register, can be used to accomplish the emulated process while not corrupting the value of the stored damage which is the data processing system.

MR. SPOONER: Sure.

JUDGE HOMERE: Okay, so in that case, so when it's a zero, it's stored in the dummy register.

MR. SPOONER: Okay.

JUDGE HOMERE: So technically I mean so we really -- we can focus that to what I just alluded to earlier. When it's a zero, when the value key bit is a zero --

MR. SPOONER: Right, right.

JUDGE HOMERE: -- the result is stored in the dummy register, whereas when it's a one, it's stored in the data processing register.

MR. SPOONER: I have no problem with that. The problem -- that's not inconsistent with our position, because it doesn't teach the fact that this condition code is embedded in the write code. This is part of the -- these are their -- the bits are part of their public or private key that they're referring to. That's not --

JUDGE HOMERE: Well, but this is not transparent to the user. The user does not know that this is happening, the attacker -- this whole process is taking place in order to fool the attacker, so the attacker doesn't know that it's something that's happening privately in the system.

MR. SPOONER: But the attacker will see that. He will see that in the power signature analysis, the power and timing analysis of how that circuit works. In other words --

JUDGE HOMERE: That's the whole purpose --

MR. SPOONER: -- you lose your smart card, and somebody wants to get money out of your ATM. They throw it into an analyzer, and they put signal inputs into it. They look at then the output . What happens? And so by putting in a whole sequence of different signals, timing, power levels and all of that, they will get different outputs. By doing that repeatedly, that's the differential power analysis, even if QIU is built into your smart card system, they will ultimately see the fingerprint of the branching that occurs in the algorithm that's used to protect that smart card, and they will then be able to deduce what your key is and use that card and get money out of your ATM.

JUDGE HOMERE: At the very outset that's why -- that's the exact problem that QIU is trying to resolve, and it says that right here in column 1, lines --

MR. SPOONER: We agree.

JUDGE HOMERE: Yes.

MR. SPOONER: We agree completely.

JUDGE HOMERE: Okay.

MR. SPOONER: Both systems are trying to protect that smart card. QIU protects against the simple power analysis. If you look at the bold view, you'll see that QIU, by storing to the dummy register -- remember, we never said that QIU doesn't have a trash register. QIU does, and it stores to that trash register so that the overall broad power, if you look at the broad power requirements of that chip, it will look very similar, and you won't be able to figure it out. But if you do it many, many times, and you look very closely at it, you'll see the difference in storing, the difference which occurs in that branching at -- of the algorithm that occurs when you decide to go to the trash register as opposed to the other. And in our invention, we don't do that. We don't have that branching determination that is made, so you can't see it. It's encoded in the instruction, in these conditional-write data processing instructions, and that's the trick. That's why this thing is -- both of them defeat power analysis. QIU only defeats the simple power analysis. Ours defeats both the simple power analysis and the more advanced differential power analysis, differential being looking at a bunch of different times of usage with the same inputs to see how this thing branches and then figuring out which branch of the algorithm takes place and then using that information to --

JUDGE HOMERE: I have a residual question for you.

MR. SPOONER: Okay.

JUDGE HOMERE: What language here in this claim that captures the concept of a differential power analysis here?

MR. SPOONER: There is nothing that says differential power analysis. There's no wording in the claim or in the specification or anything else. That's what --

JUDGE HOMERE: And that's what I'm saying, what language? I mean do you have any language in there that would --

MR. SPOONER: The language that enables differential power analysis to be defeated is the language up in the third paragraph that says at least one data processing instruction executed though the -- is a conditional-write data processing instruction encoding condition codes, specifying conditions under which conditional-write data processing will or will not be permitted to write data to effect a change.

JUDGE HOMERE: Okay, I think we understand each other.

JUDGE HUGHES: If I understand your argument correctly --

MR. SPOONER: And I'm not sure I understand it correctly, so feel free to ask.

JUDGE HUGHES: Okay, well, we're going to hold your feet to the fire here because, you know, we're giving lots of names to different things, but we're not getting down to the meat of this claim. What I'm understanding your argument to be is that the instruction itself has a code in it, a zero or a one, as opposed to you're saying QIU --

MR. SPOONER: Not necessarily at --

JUDGE HUGHES: -- does not have that instruction with that --

MR. SPOONER: Not necessarily a zero or a one, but otherwise you're correct.

JUDGE HUGHES: Well, then some kind of code.

MR. SPOONER: Yes.

JUDGE HUGHES: You don't specify what kind of code, so some kind of coding in it. If you --

MR. SPOONER: Right, well, it does -- it specifies conditions under which conditional-write data processing will or will not, so it is some code that specifies those --

JUDGE HUGHES: Right, so will or will not is digital. It's a zero or one, so can we just get down to --

MR. SPOONER: Well, it could be.

JUDGE HUGHES: -- that it either tells it to do -- it's conditional? It either does or doesn't do something --

MR. SPOONER: Right.

JUDGE HUGHES: -- and a code that tells it to do something or not do something and QIU --

MR. SPOONER: But it might tell it write, always write. Write to this one in --

JUDGE HUGHES: All right, I'm trying to get you to simplify your argument here a little bit. Would you agree that QIU has a -- has instructions that write to a dummy register or do not write to a dummy register as part of its algorithm?

MR. SPOONER: I don't think QIU has instructions as that term is commonly understood.

JUDGE HOMERE: It doesn't like the term instruction.

JUDGE HUGHES: An algorithm --

MR. SPOONER: An algorithm, yes.

JUDGE HUGHES: All right, and an algorithm -- you're saying an algorithm isn't made up of instructions then? Because that's how I understand what an algorithm --

MR. SPOONER: Algorithm is different from instructions.

JUDGE HUGHES: Okay. Well, we might have to agree to disagree on that point. But --

MR. SPOONER: Take the definition of instructions. We attached a copy to the Reply Brief. Examiner never saw it, so take it for what it's worth.

JUDGE HUGHES: Okay.

MR. SPOONER: Or do your own Internet search.

JUDGE HUGHES: So going back again to your explanation seems to be that the invention requires an instruction with encoding to do a conditional operation, either write or not write?

MR. SPOONER: Yes.

JUDGE HUGHES: And you're saying QIU does not have instructions that are so encoded?

MR. SPOONER: Right, exactly. That's the first thing we're saying QIU doesn't have.

JUDGE HUGHES: Okay.

MR. SPOONER: The second thing we're saying QIU doesn't have is the trash register as defined in the claim. Not QIU has a trash register, but he doesn't have a trash register that has any of the other attributes, interrelationship attributes specified in the claim to which a data -- result data value will be written instead of a data processing register. Upon execution of said conditional-write data processing instruction, when said condition codes within said conditional-write data processing instruction do not permit a write to effect a change in the state of the processor core. He doesn't have that either.

JUDGE HOMERE: One question.

MR. SPOONER: Okay.

JUDGE HOMERE: Okay, well, the zero -- let's focus on the zero and on the binary status of QIU. QIU says that well, if you have a zero -- I'll just -- we'll go on that. If you have a zero, then you write to a trash register. If you have a one, then you do not write to the trash register.

MR. SPOONER: You're talking about this column 6 now?

JUDGE HOMERE: Column 6, yes.

MR. SPOONER: Okay.

JUDGE HOMERE: If you have a zero, you write to the trash register. If you do not have a zero, then you're not permitted to write to the trash register.

MR. SPOONER: Okay.

JUDGE HOMERE: Right. Now isn't it -- aren't these conditions that determine whether -- when it is acceptable to write to the trash register or not?

MR. SPOONER: Sure, sure.

JUDGE HOMERE: How would that be --

MR. SPOONER: They are conditions, no doubt. But that's not the same as the condition code, an encoded condition code. Different animal.

JUDGE HOMERE: Okay. All right. Do you have anything else?

MR. SPOONER: Well, just to diverge somewhat, if we don't -- if QIU doesn't teach this data processing instruction with its encoded conditional code and if it doesn't -- or if it doesn't teach a trash register which is responsive to these things, as set out in the claim, it can't possibly anticipate. If it can't anticipate, then combining that reference with another reference in the obviousness rejection, that doesn't support the obviousness argument as well, because the Examiner doesn't allege that the at least one processing instruction or the trash register as claimed are shown in the second reference. And so the first prong of the obviousness test isn't met. Second prong is there's no reason to combine them. Third prong is and our contention is that the prior art teaches away from it. It precludes it. So there is no anticipation. There is no obviousness. But it all hinges on whether those last two paragraphs of claim 1 and similar method steps in claim 6 are shown in QIU, and they're just not there. QIU is an algorithm based system. Ours is a write instruction based system.

JUDGE HOMERE: I think that --

MR. SPOONER: Thank you so much.

(Whereupon, the proceedings were concluded on Wednesday, November 4, 2009.)